WEB

第 5 章　Servlet 技术

本章目录

前言： Servlet 的定义

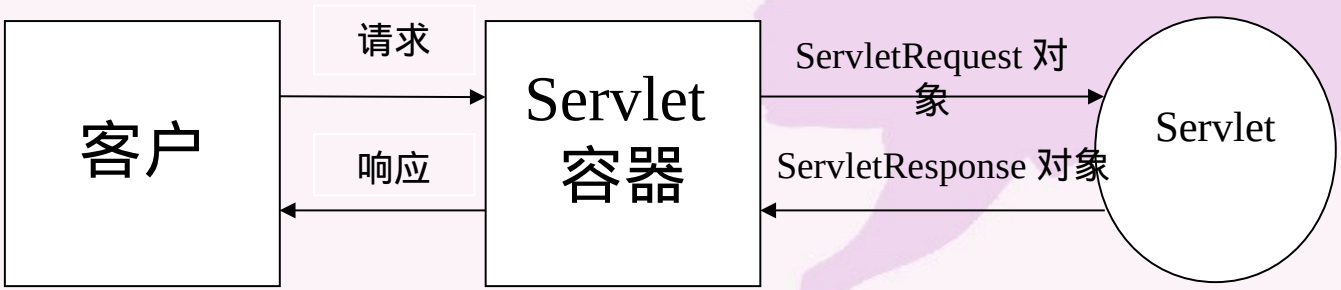
- 所谓 Servlet ，首先是一个 Java 类，是一种运行在支持 Java 应用服务器上的 Web 组件，它与普通 Java 类的区别在于它是一个专门处理请求和响应的 Java 类。
- Servlet 是成熟的 J2EE （ Java 2 Enterprise Edition ）组件技术之一。 Servlet 属于目前主流的 MVC （ Model - View - Controller ）架构中的控制器层（ Controller ）组件，主要功能是接受前端用户的请求，调用后端的逻辑处理程序，最终给客户返回

本章将着重介绍 Servlet 的工作原理、常用的 Servlet API 以及如何通过 MyEclipse 快速开发 Servlet 程序。

5.1 Servlet 的工作原理

5.1.1 Servlet 容器

Servlet 容器属于 Java 应用服务器的概念范畴，其作用是负责处理客户请求，当客户请求来到时， Servlet 容器获取请求，然后调用某个 Servlet ，并把 Servlet 的执行结果返回给客户



5.1 Servlet 的工作原理

5.1.2 Servlet 的生命周期

一个 Servlet 具有一个生命周期，这个生命周期定义了一个 Servlet 如何被载入并被初始化，如何接收请求并作出响应，如何从服务器中被清除。

所有的 Servlet 都会直接地或间接地执行 javax.servlet.Servlet 接口，这样它才能在一个 Servlet 引擎中运行。 Servlet 引擎是 Web 服务器按照 Java Servlet API 定制的扩展。 Servlet 引擎提供网络服务，能够理解 MIME 请求，并提供一个运行 Servlet 的容器。

javax.servlet.Servlet 接口定义了在 Servlet 的生命周期中特定时间以及特定顺序被调用的方法。

servlet 的生命周期如下：

1. Servlet 容器创建 servlet 的一个实例
2. 容器调用该实例的 init 方法进行初始化。
3. 当客户端向该 Servlet 发送请求时，容器调用此实例的 service 方法
4. 在 service 方法中，根据当前用户请求的方式进一步调用 doGet 或者 doPost 方法进行处理。
5. 当 Servlet 容器终止运行或 Servlet 容器重新装载 Servlet 的新实例时， Servlet 容器调用 Servlet 的 destrov 方法释放 Servlet 所占用的资源。

一旦客户端请求一个 Servlet ，容器必将执行一个完整的 Servlet 生命周期。

容器在 Servlet 首次被调用时创建它的一个实例，并保持该实例在内存中，让它对所有的请求进行处理。若容器关闭重启或者 Servlet 类内容产生变化，则这个内存中的实例就会被销毁并在下一次被请求时重新被创建。

5.2 Servlet API

- Servlet 容器可以创建实现 ServletRequest 和 ServletResponse 两个接口的对象
- Servlet 应用程序编程接口，即 Servlet API（ Application Programming Interface ）包含 SUN 公司提供的两个软件包：

  javax.servlet 包定义了所有的 Servlet 类都必须扩展的通用接口和类；

  javax.servlet.http 包定义了采用 HTTP 协议通信的 HttpServlet 类以及相关接口。

几个常用的接口和类：

1.javax.servlet.http.HttpServlet

该接口是 Servlet API 的核心类，我们自定义的 Servlet 都是该类的子类

（1）int 方法

负责初始化 Servlet 对象。

（2）doGet 和 doPost 方法

GET 方式，直接在浏览器地址栏输入 URL 按回车，即向服务器发送了一次 GET 方式的请求； POST 方式，通常是客户端通过一个表单把信息提交给服务器。

（3） service 方法

service 方法负责处理当前客户的请求，为用户提供服务。

（4）destroy 方法

负责释放 Servlet 对象占用的资源。该方法仅执行一次，即在 Servlet 容器停止执行该方法。

2.javax.servlet.http.HttpServletRequest

HttpServletRequest 接口包含了客户端请求的信息，可以通过实现该接口的对象取得客户端的一些信息（例如表单数据、客户端 IP 地址等）

（1）getParameter 方法

获取客户以 GET 方式发送的参数或者以 POST 方式提交的 form 表单的控件值。如果有多个同名的控件或参数，则返回其中的第一个值。

（2）getParameterValues 方法

如果有多个同名的控件或参数，则该方法可以获得对应的多个参数值，并且放到一个 String 数组里，比如复选框集（ checkbox ）。

3.javax.servlet.http.HttpServletResponse

Servlet 容器提供一个实现 HttpServletResponse 接口的对象并通过 service() 方法将该对象传递给 Servlet。 Servlet 通过该对象及其方法可以向客户端浏览器返回结果。

（1）setContentType 方法

在给调用者发回响应前，必须用 setContentType 方法来设置 HTTP 响应的 MIME 类型，通常设置如下：

```
response.setContentType("text/html;charset=gb2312");
```

（2）getWriter 方法

该将返回 PrintWriter 对象，把 Servlet 的结果作为文本返回给调用者。 PrintWriter 对象自动把 Java 内部的 Unicode 编码字符转换成正确的编码以使客户端能够阅读。

（3）getOutputStream 方法

getOutputStream 方法返回 ServletOutputStream 对象，它是 java.io.OutputStream 的子类。此对象向客户发送二进制数据。

5.3 Servlet 的开发步骤

开发 Servlet 程序的典型步骤如下：

1. 创建自定义的 Servlet 类，继承父类 HttpServlet。
2. 在自定义的 Servlet 中覆盖父类 HttpServlet 的部分方法，如 doGet() 或 doPost() 方法，对其方法体进行重写。
3. 在 web.xml 中为 Servlet 进行注册。
4. 获取 HTTP 请求信息，例如从 HttpServletRequest 对象中获取客户端提交的参数，可以通过 getParameter(String name) 方法进行获取。
5. 生成 HTTP 响应结果。通过 HttpServletResponse 对象可以生成响应结果。此外，还可以使用 HttpServletResponse 对象的 sendRedirect() 方法重定向到其它 URL。

5.4 Servlet 的开发实例

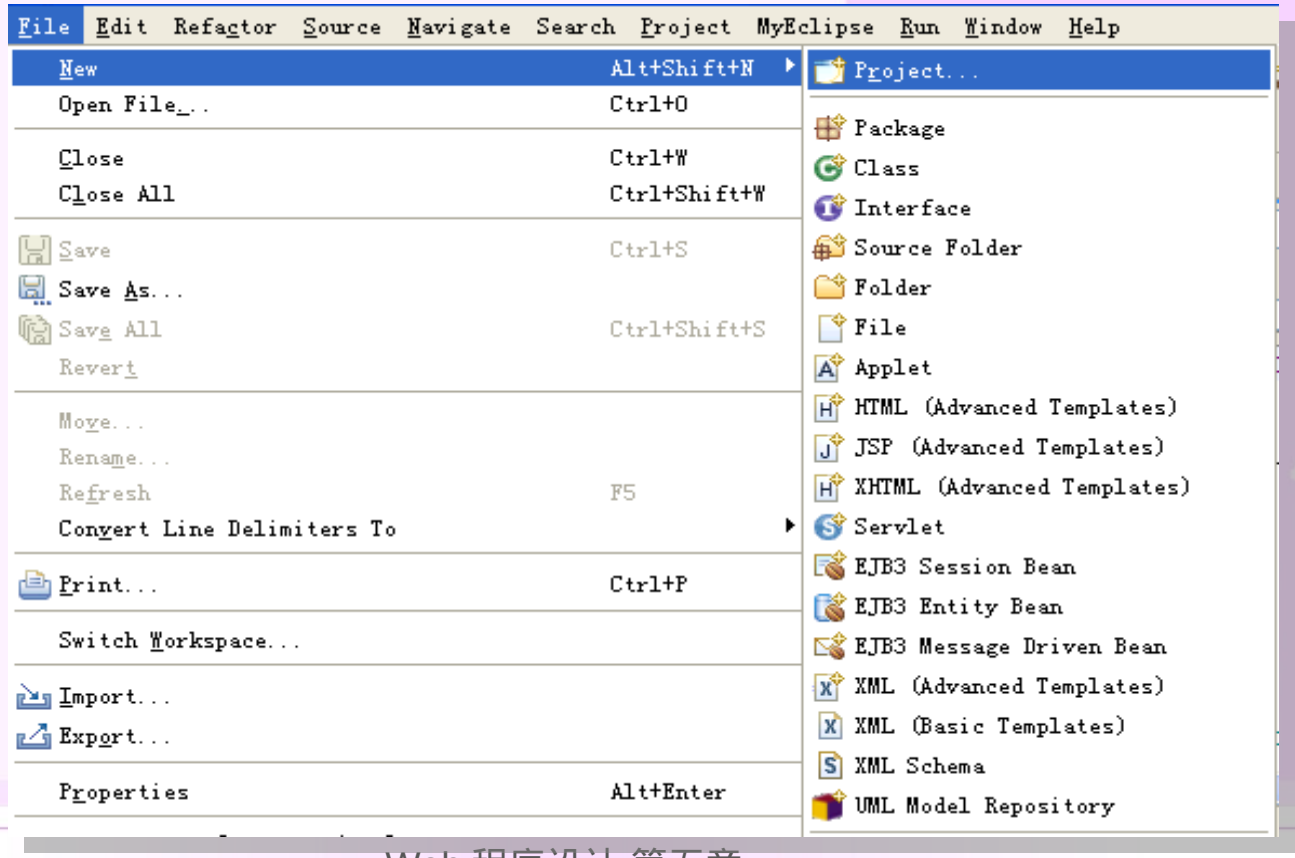
- Servlet 作为控制器，可以接收并处理用户的请求
- Tomcat 服务器调用 Servlet 的 doGet 方法处理客户端通过地址栏传递过来的参数（如用户名和密码），并将它们打印出来
- 在获取 request 对象中的参数时，如果遇到中文字符串且不做任何处理，则显示出来的将会是乱码，比如本例中的“用户名”有可能出现中文字符，就必须进行中文编码转换才能正常显示

GET 请求方式：

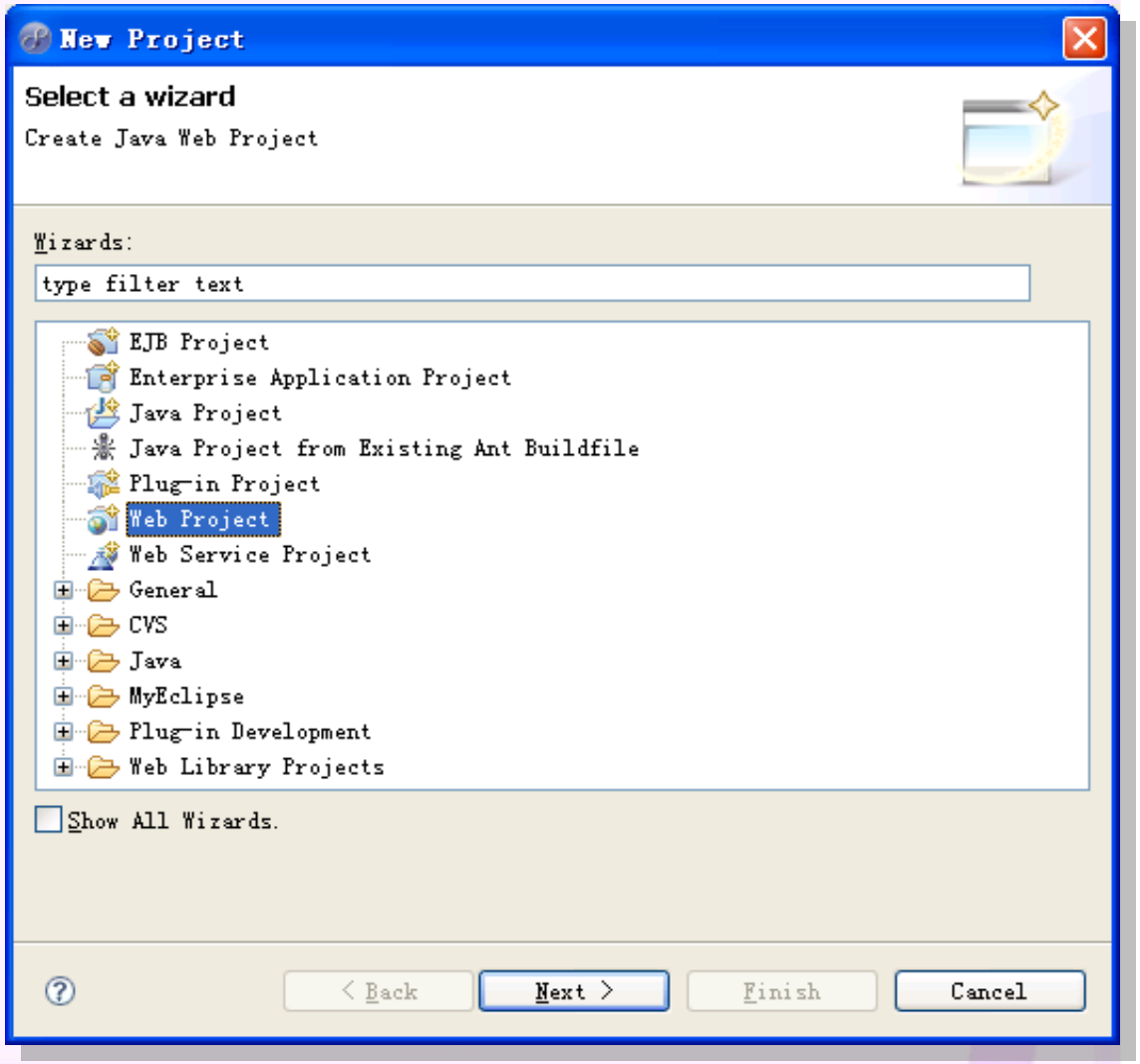
URL ： http://localhost:8080/hello/login?username= 小王 &password=123

下面我们将开发 loginServlet 类，具体步骤如下：

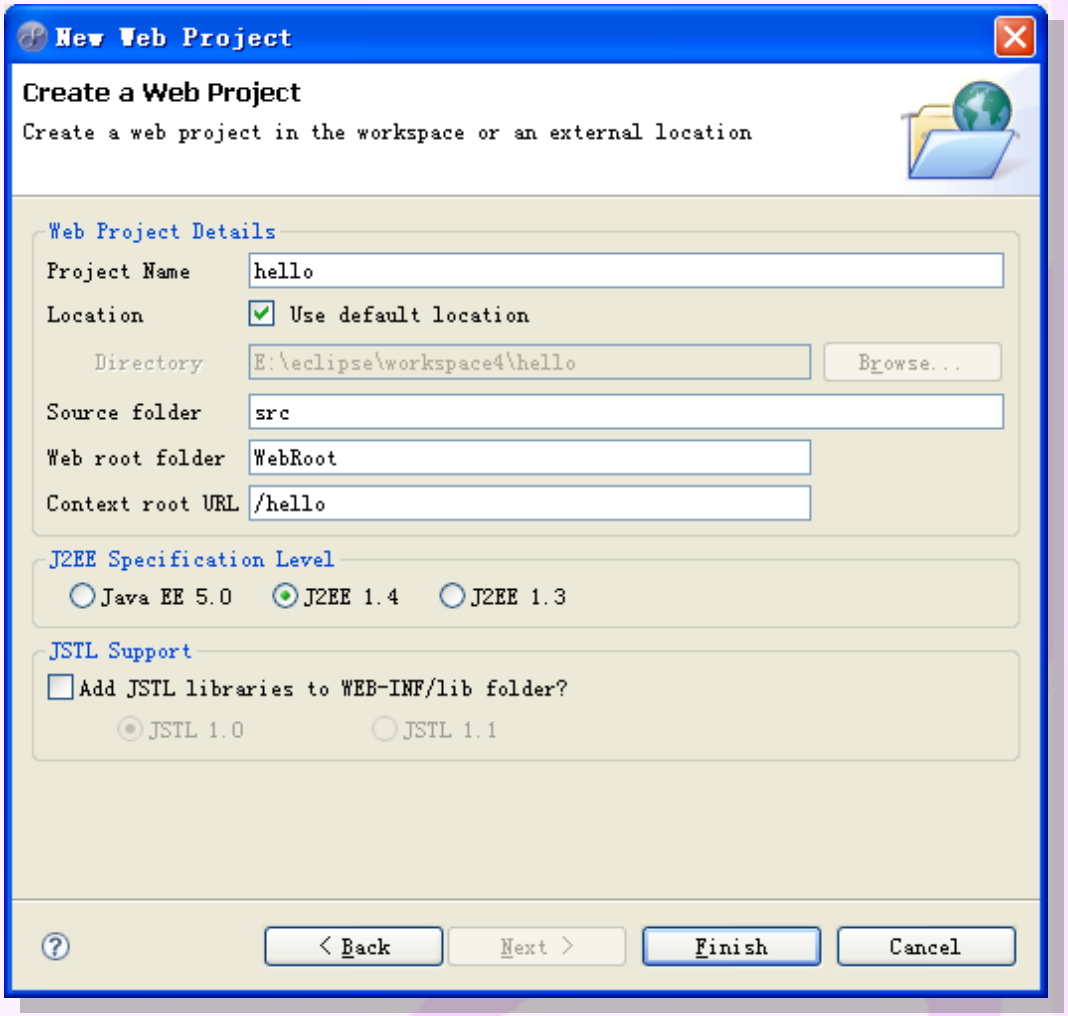
- 1. 在 MyEclipse 开发工具中选择【 File 】→【 New 】→【 Project 】，新建一个项。如图 5-3 所示：



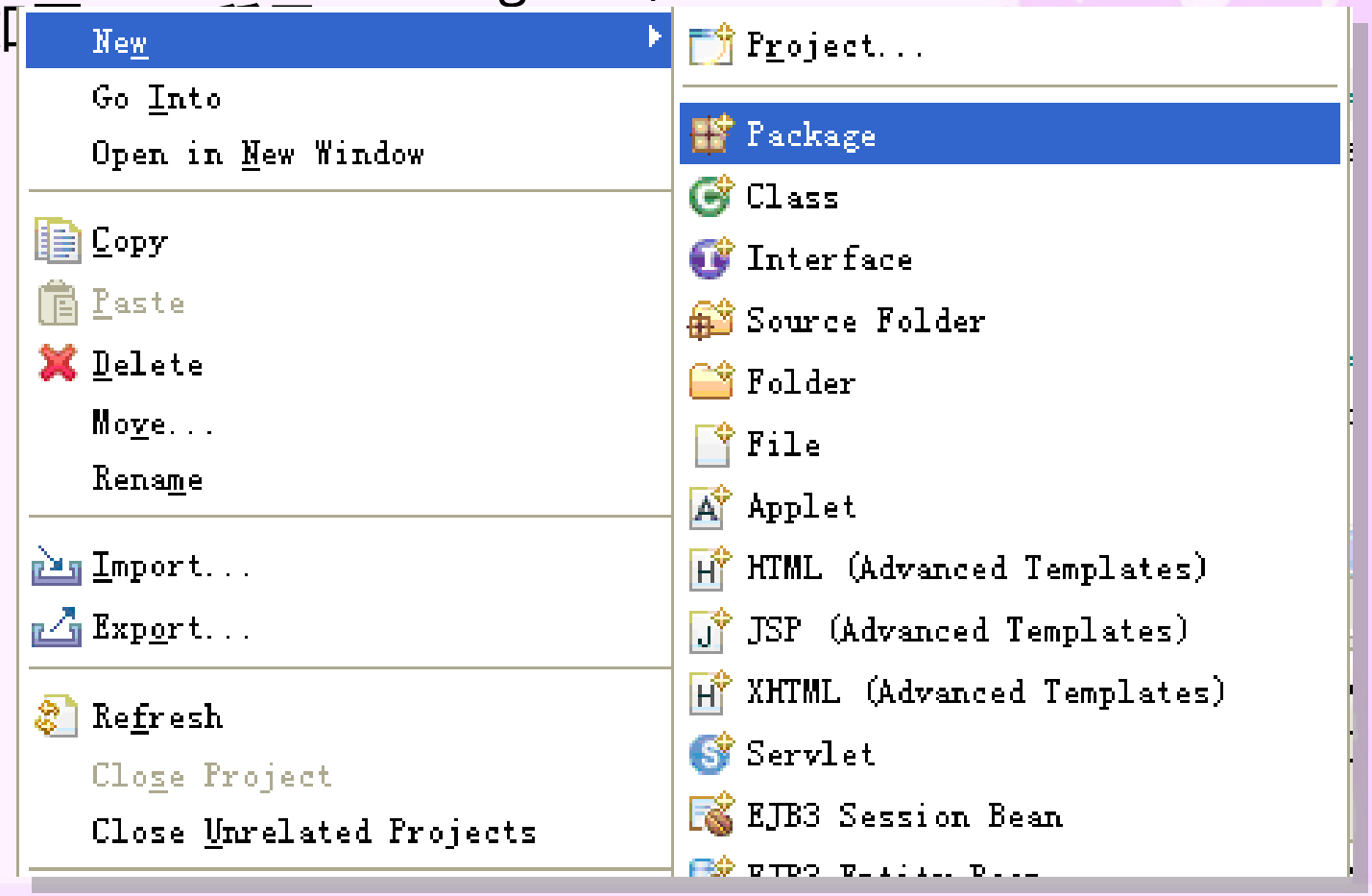
- 2. 选中后系统会弹出如图5-4 所示的项目创建向导对话框，我们选中“ Web Project” 选项



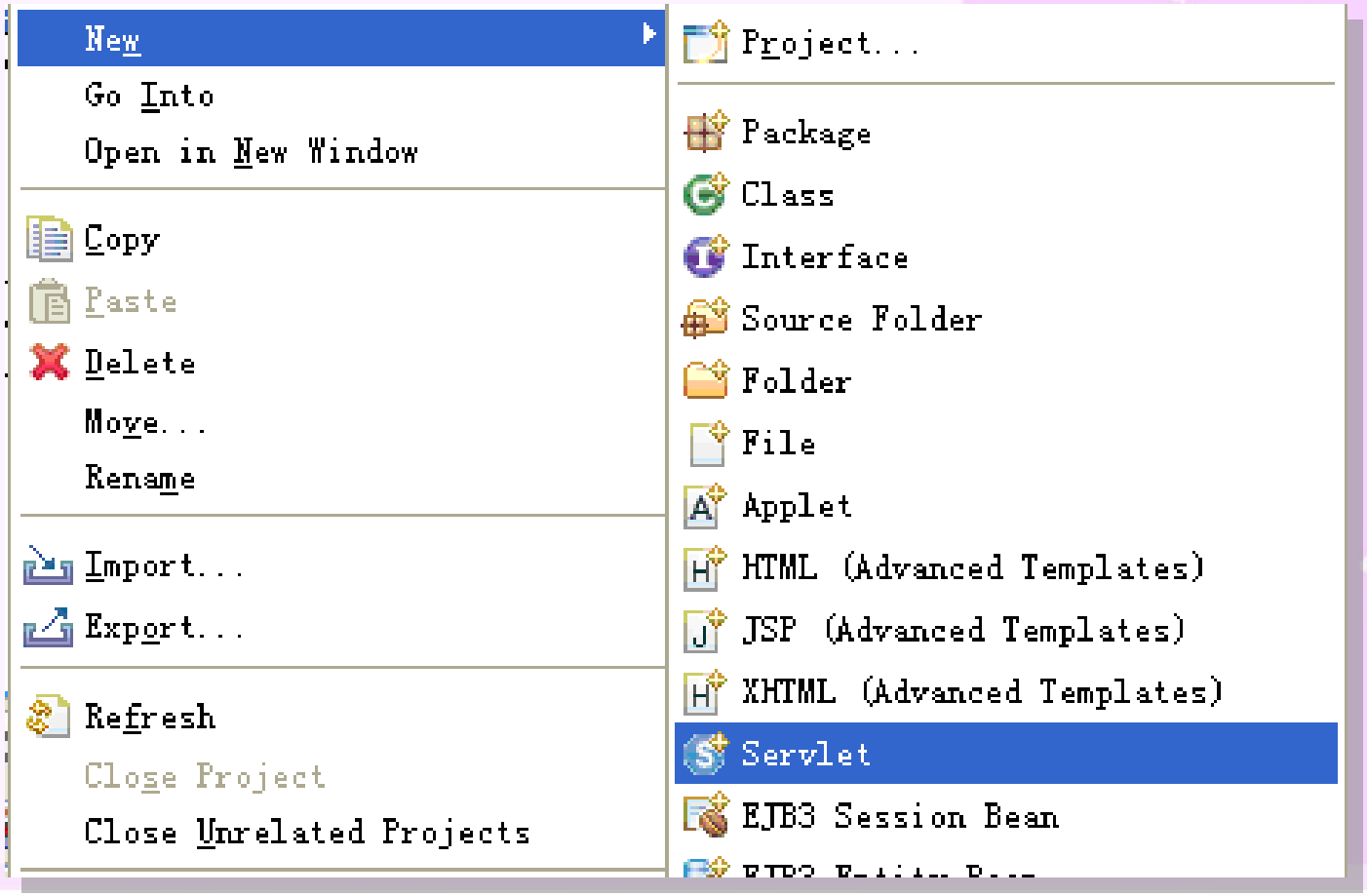
- 3. 单击【 Next 】按钮，弹出如所示的对话框，此处我们给当前的 Web 项目命名为“ hello”



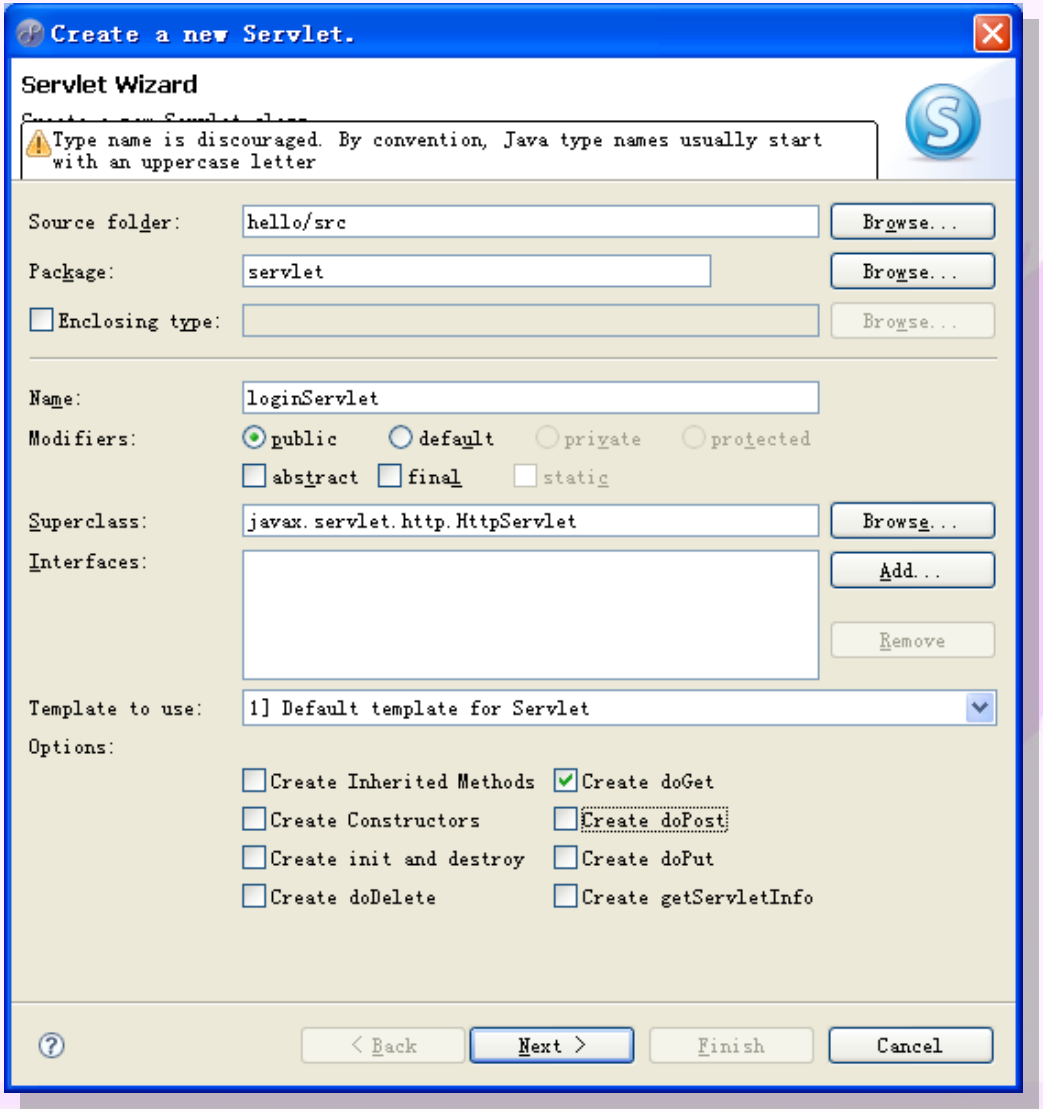
- 4. 单击【 Finish 】按钮。完成后，展开 hello 项目，右键单击 src 文件夹，在弹出菜单中选择【 New 】→【 Package 】，新建一个名为 Servlet 的包。如



- 5. 右键单击 Servlet 包，在弹出菜单中选择【 New 】→【 Servlet 】。如图 5-7 所示：



- 6. 弹出如图5-8 所示的Servlet 创建向导
- 将此其命名为loginServlet
- 父类默认是HttpServlet
- 在复选框中选择需要重写的父类中的方法
- 这里我们选择“Create doGet”方法。



- 7. 单击【 Next 】按钮，弹出如图 5-9 所示的 Web.xml 配置向导：

Servlet 的在 web.xml 中的配置

- web.xml 文件存放于 Webroot 的子目录 WEB-INF 下。 MyEclipse 的 Web.xml 的配置向导可以将该 Servlet 的映射信息注册到 web.xml 文件中。
- 为了将 Servlet 的各配置属性严格区分开，我们将系统自动生成的“ Servlet/JSP Name” 属性值改为“ loginServlet” ，“ Servlet/JSP Mapping URL” 属性值改为“ /login” 。
- 以下是配置完成后 web.xml 文件的代码：

```
<servlet>
……
<servlet-name>loginServlet</servlet-name>
    <servlet-class>servlet.loginServlet</servlet-class>
</servlet>
  <servlet-mapping>
    <servlet-name>loginServlet</servlet-name>
    <url-pattern>/login</url-pattern>
 </servlet-mapping>
……
<welcome-file>index.jsp</welcome-file>
```

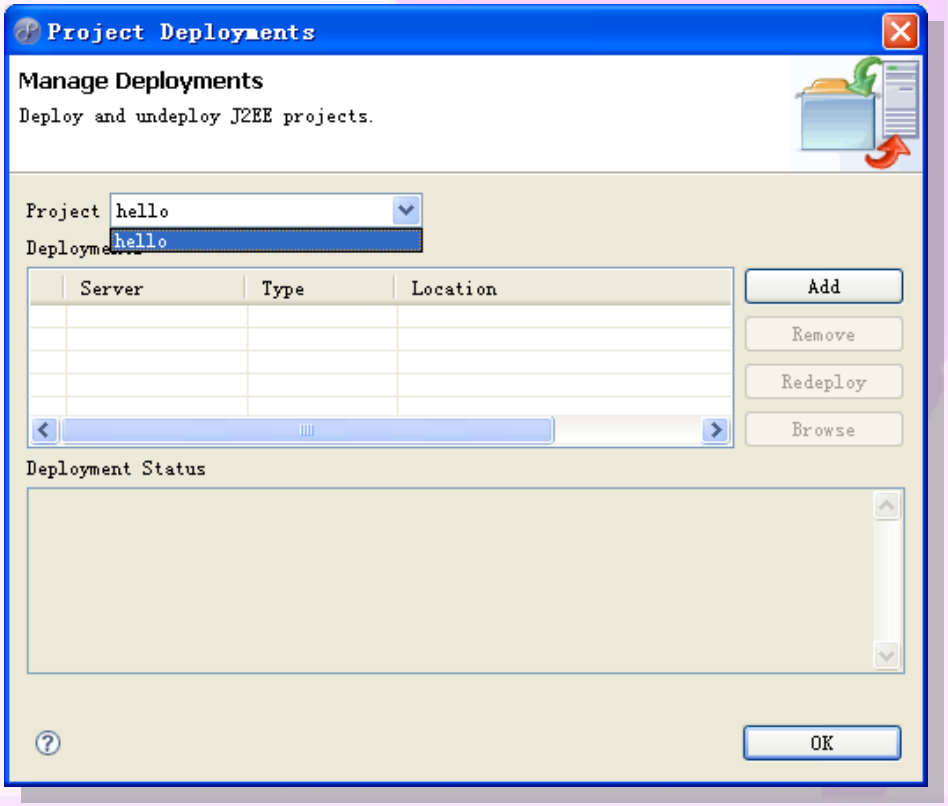
以上 web.xml 中的 <welcome-file-list> 以及子元素 <welcome-file> 的配置信息是 MyEclipse 为每一个 Web 项目自动生成的，其作用是将 index.jsp 设置为默认首页

以下是 loginServlet 中 doGet 方法的实现代码

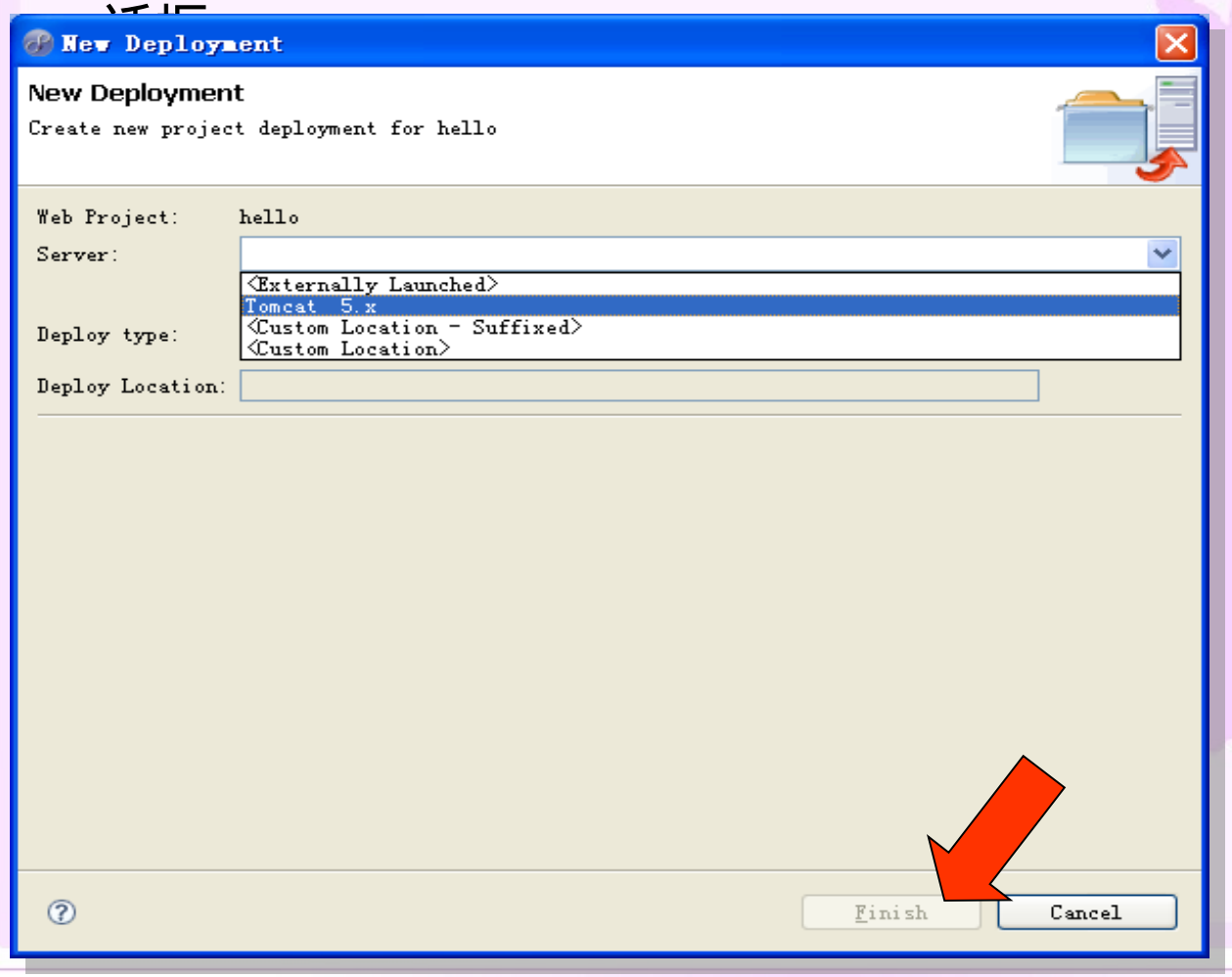
```
public void doGet(HttpServletRequest request, HttpServletResponse
response)
throws ServletException, IOException {
        String username=request.getParameter("username");
        String password=request.getParameter("password");
         response.setContentType("text/html;charset=gb2312");
         PrintWriter out = response.getWriter();
         if (username != null) {
            out.println(" 您的用户名是： "+username+"<br>");
         }
         if (password != null) {
            out.println(" 您的密码是： "+password);
         }
         out.flush();
         out.close();       }
```

- 完成以上代码后，在 MyEclipse 的工具栏上单击项目发布图标，如图 5-10 所示：

- 单击该图标后，系统会弹出如所示的项目发布对话框：



- 在“ project” 下拉框中选择所要发布的项目，本例为“ hello” 。然后单击【 Add 】按钮，弹出如图 5-12 所示的服务器选择对



选定后，单击【 Finish 】按钮完成整个项目的部署。

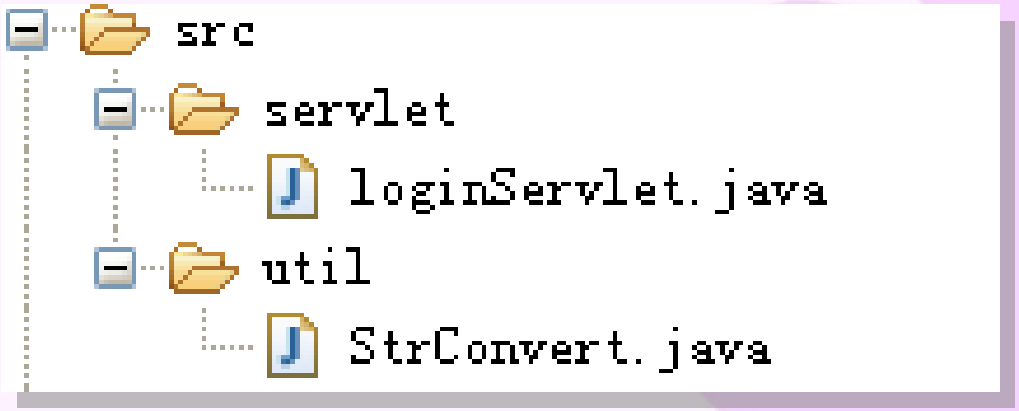
- 输入中文显示乱码问题

  由于地址栏中的默认字符集编码是“ ISO-8859-1”，因此对于用户用 get 方式提交的中文字符串参数必须先按照“ ISO-8859-1” 的编码还原成一个 byte[] 字节数组，再用 String 类的构造函数 String(byte[] bytes,String charsetName) 将其转换为支持中文字符集编码的对象。具体代码如下：

```
String tempStr=request.getParameter("username");
Byte tempbyte[]=tempStr.getBytes("ISO-8859-1");
tempStr=new String(tempbyte,"gb2312");
```

为了在其它也需要处理中文字符串的程序中可以重用以上代码，我们可以将以上处理代码封装成模型层中的方法 .

- 为了与 Servlet 程序区分开，我们在当前项目的“ src”下创建一个名为“ util”的包，在“ util”包下创建一个名为 StrConvert 的类，类包文件的结构如图 5-13 所示：



在 StrConvert 类中，封装了中文字符串处理的静态方法，具体实现如下：

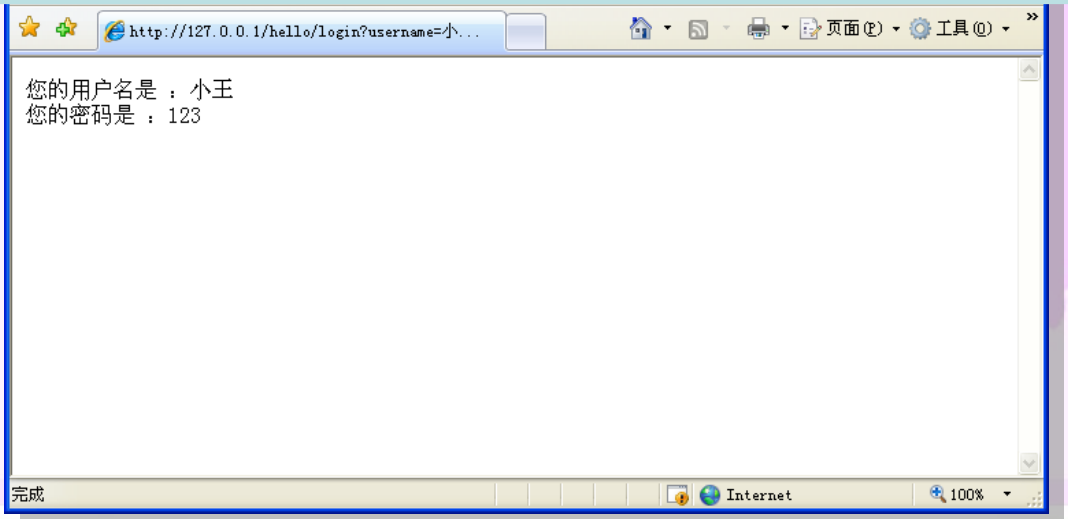
```
public static String ToCN(String str)
{
    String strcn=null;
    try {
  strcn=new String(str.getBytes(“iso-88591"),"gb2312");
    }
   catch (UnsupportedEncodingException e)
    {
        // TODO Auto-generated catch block
     e.printStackTrace();
    }
    return strcn;
}
```

- 完成 StrConvert 类的编写后，需要修改 loginServlet 的 doGet 方法，在获取参数后调用 StrConvert 的 ToCN 方法进行中文字符串的处理，代码如下：

```
String username =
StrConvert.ToCN(request.getParameter("username"));
```

再次在浏览器中输入以下地址，得到页面如图 5-14 所示：

http://localhost:8080/hello/login?username= 小王 &password=123



5.5 本章小结

MVC 框架中控制器 Servlet 的相关概念，包括 Servlet 容器、 Servlet API 以及 Servlet 的生命周期等。在 Servlet API 部分重点介绍了 HttpServlet 类、 HttpServletRequest 接口以及 HttpServletResponse 接口的一些常见的实用方法。在最后的开发实例中详细介绍了如何使用工具 MyEclipse 快速开发一个以 get 方式请求 Servlet 的应用程序。本章的内容对于学习 Java Web 技术的读者而言较为重要，有关 Servlet 与 JSP 等其它 Web 组件结合使用的内容还会在后续章节中再做详细介绍。

本章习题：

1. Servlet 在 MVC 框架中的作用是什么？
2. 简述 Servlet 的生命周期及其工作原理。
3. Servlet 完整类名是什么？有哪些常见的方法？
4. 在 Servlet 相关的 API 中，请求和响应接口分别是什么？有哪些常见的方法？
5. 请开发一个 Servlet 程序，要求能够获取客户端发送的用户名和密码，并判断用户名和密码是否分别为“ admin” 和“ 123456” ，最终在浏览器中打印是否验证通过的信息提示。